Software to convert terrestrial LiDAR scans of natural environments into photorealistic meshes

Software to Convert Terrestrial LiDAR Scans of Natural Environments Into Photorealistic Meshes

Risse, Benjamin[a,*], Mangan, Michael[b], Stürzl, Wolfgang[c], Webb, Barbara[a]

[a] *School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh, EH8 9AB, UK*
[b] *Lincoln Centre for Autonomous Systems, University of Lincoln, Brayford Pool, Lincoln, LN6 7TS, UK*
[c] *Institut für Robotik und Mechatronik, Deutsches Zentrum fur Luft- und Raumfahrt (DLR), Münchener Strasse 20, 82234 Oberpfaffenhofen-Wessling, Germany*

Abstract

The introduction of 3D scanning has strongly influenced environmental sciences. If the resulting point clouds can be transformed into polygon meshes, a vast range of visualisation and analysis tools can be applied. But extracting accurate meshes from large point clouds gathered in natural environments is not trivial, requiring a suite of customisable processing steps. We present *Habitat3D*, an open source software tool to generate photorealistic meshes from registered point clouds of natural outdoor scenes. We demonstrate its capability by extracting meshes of different environments: $8,800\text{m}^2$ grassland featuring several Eucalyptus trees (combining 9 scans and $41,989,885$ data points); $1,018\text{m}^2$ desert densely covered by vegetation (combining 56 scans and $192,223,621$ data points); a well-structured garden; and a rough, volcanic surface. The resultant reconstructions accurately preserve all spatial features with millimetre accuracy whilst reducing the memory load by up to 98.5%. This enables rapid visualisation of the environments using off-the-shelf game engines and graphics hardware.

*Corresponding author
Email address: brisse@inf.ed.ac.uk (Risse, Benjamin)

Software Name	Habitat3D
Developer	Benjamin Risse
Contact	brisse@inf.ed.ac.uk
Affiliation	University of Edinburgh (IPAB)
Available since	2017
Operating System	Linux, Mac, Windows
Programming language	C++
Requirements	Qt, PCL, VTK, BOOST
Availability	insectvision.org
Costs	Free (open source)
Dataset 1	Canberra
Availability	insectvision.org
Format	PLY
Size	470Mb
Dataset 2	Seville
Availability	insectvision.org
Format	PLY
Size	941Mb
	For details see Appendix A.2

1. Introduction

Recent technological advances, specifically the availability of commercially priced Light Detection And Ranging (LiDAR) scanners, have been instrumental in recent efforts to accurately map natural environments (for reviews see [1, 2, 3, 4]). Historically, airborne LiDAR scanners were used in combination with aerial or satellite imagery to build large scale digital terrain maps and extract environmental properties such as canopy surface topography, leaf area index, and above-ground biomass for applications ranging from forestry to natural resource management and geomorphology (for a review see [5]). More recently, the sub-field of proximal remote sensing has emerged whereby smaller areas of particular interest are mapped in great detail using terrestrial scanners. We are particularly interested in the use of this approach in ethological studies, for example the insights into the nesting habitats of birds [6], the selection of kill sites by lions [7] and the navigation behaviours of bats [8] and wasps [9, 10], with important implications for conservation, ecology and forest management among others.

The common requirements of the latter studies is to obtain an accurate 3D description of the environment of interest. However the dataset returned by LiDAR, raw point clouds, can be huge, noisy, highly redundant and difficult to interpret. Point clouds do not reflect the underlying topology (connectedness of the points) so useful processes such as distinguishing surfaces and objects, adding semantic labels, and estimating volumes are difficult. In addition, visualisation of point clouds is slow and non-intuitive, e.g., with sparseness of the points increasing with zoom. In particular, there is little possibility to manipulate the appearance, e.g., removing objects, enhancing appearance of surfaces by adding pattern, texture or transparency, altering the lighting conditions, or to provide physical constraints in the viewer's interaction with the scene, such as preventing penetration of surfaces (collision control).

As a consequence it is widely accepted that there are many benefits if a point cloud can be converted to a more standardised polygon model (mesh) represen-

tation (e.g. [11, 12]). Primarily, this allows exploitation of existing optimised hardware and software for visualisation, such as game engines and virtual reality. A mesh description of the 3D structure is likely to be more compact, can have direct semantics applied, and can be easily manipulated. However, producing a mesh from LiDAR scans remains a non-trivial process, particularly when a large number of high resolution scans have been taken of a natural scene, as is frequently the case in remote sensing applications. For example, in our specific target application, the aim was to reconstruct an 1018m^2 area of desert ant habitat, including uneven terrain and over 1000 plants, from 56 laser scans producing almost 200 million data points, as a mesh that could be used in a virtual world to reproduce the visual experience of a navigating ant. Existing tools such as the open source programs Meshlab could not process even a reduced subset of this data and other tools like CloudCompare does not provide the necessary processing routines. Also commercially available software such as Agisoft Photoscan or RealityCapture were not optimised for our purposes.

Well-developed methods already exist for modelling outdoor environments in specific settings and different applications (e.g. urban enviroments [13, 14], tree structures and forestry [15, 16, 12, 17, 18], plants and leaves [19, 20], wood volumes [11, 21, 22], archeological sites [23], robotic applications [24], geomorphology [25]). However, these do not generalise to the large natural scenes of interest in remote sensing.

Successful transformation of point cloud data to a usable mesh requires a number of processing steps, including: merging of scans; splitting and sampling of point clouds to facilitate computation; multiple forms of filtering, feature extraction, and clustering; the transformation to a mesh itself; and post-processing to improve the resulting model. To complicate this process, different measurements in different environments from different scanners need different combinations of these processing steps. As processing of point clouds is very time consuming, a recipe-like pipeline for batch processing is necessary; and as the steps need to be fine-tuned (e.g. choosing parameters) for the particular task and data, user feedback at each stage is needed (i.e. visualisation in a GUI).

Finally the framework needs to be able to handle the gigabytes of data that are produced in typical remote sensing applications using modern laser scanners.

Here we present the first generic open source framework incorporating all the above necessary requirements to extract complete and photorealistic meshes of natural outdoor scenes using multiple point clouds. A key contribution is that we have identified and evaluated a unique pipeline incorporating a variety of different processing steps that are critical to the problem of meshing point clouds that include dense vegetation. This pipeline is successfully used to reconstruct four qualitatively different datasets. We have incorporated and adjusted all the necessary processing, filtering, clustering and meshing algorithms into a single framework greatly increasing accessibility of LiDAR processing tools for non-experts. The resulting tool, *Habitat3D*, provides an interactive pipeline for batch processing data with informative feedback for all steps. We demonstrate its usability by evaluating different animal habitats with millimetre precision, representing a scale and level of detail that goes beyond the state of the art. The resulting scene description can be used in any rendering or game engine, so that both state-of-the-art hard- and software acceleration can be applied.

2. Methods

In this section we provide a complete description of our reconstruction approach, starting with the key characteristics of natural outdoor scenes that determine the overall modelling approach (c.f. Section 2.1) followed by an introduction to the data representation used throughout the paper (c.f. Section 2.2). In Section 2.3 the data collection is described in detail and Section 2.4 elaborates the pipeline applied to these datasets.

2.1. Overview of the Approach

To extract a polygon model (i.e. mesh) of a natural outdoor scene requires multiple point clouds from laser scans taken from different locations, so that objects are observed from different directions. These need to be registered

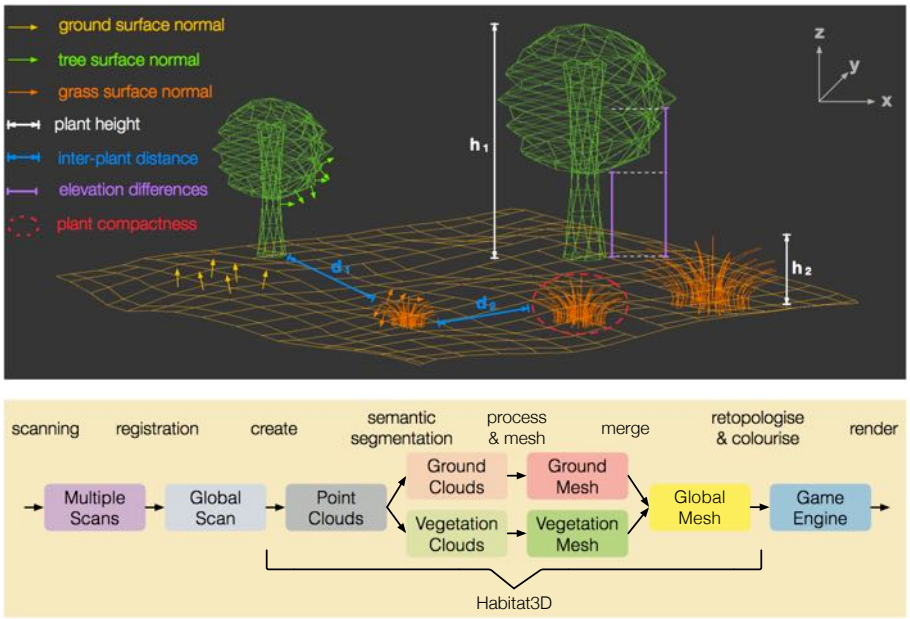


Figure 1: Modelling approach overview. Top: Simplified mesh of a natural outdoor scene. The ground is given in yellow, trees are given in green and grass bushes are given in orange. Bottom: A sequence of steps is necessary to mesh natural outdoor scenes. All processing steps done by Habitat3D are indicated.

to ensure a single global coordinate system between scans. However, a direct mesh extraction from the resultant registered clouds is not feasible, as meshing relies on features and topological assumptions that are not consistent across the variety of object surfaces in a natural scene. We thus first need to consider, from an abstract perspective, what are the available features in a natural outdoor scene that are meaningful for its modelling. Considering the polygon mesh given in Figure 1 (top), the following general characteristics can be identified:

- The curvature of the ground appears smoother than the curvature of the above vegetation implying almost no abrupt jumps in elevation.
- The surface normals on the ground (i.e. vector perpendicular to the approximated tangent plane to the ground surface) are roughly pointing up (i.e. positive along the z-axis), whereas normals along plant surfaces are

less regular and pointing in all directions.

- The underlying geometry of the ground appears as a wavy sheet whereas individual plants form a closed volume.
- The identity of different plants is given by their mutual distance and compactness of each plant (i.e. the points in the Euclidean 3D space of an individual plant subset are closed and bounded).
- Different types of vegetation can be roughly identified based on their maximal height, by different volumes (or number of points) or by variations in colour (assuming colour point clouds have been acquired).

Note that these generalisations might be affected by the spatial resolution of the scanner. We ensured the validity by altering the spatial resolution from sub-millimetre precision to strongly down-sampled test sets. All characteristics were validated under different point cloud densities indicating a robust generalisation (data not shown).

These characteristics provide the basis for the sequence of steps used in our framework (Figure 1, bottom). After registration of point clouds, the above characteristics are used to segment clouds into ground and vegetation, before the vegetation is further clustered into individual plants. The ground and vegetation clouds are processed and meshed independently to generate the respective models. Note that the steps necessary to get these models are very different (see Section 2.4). After merging the ground and vegetation the resultant global mesh is retopologised and colourised to form the overall model. This model is then ready for analysis, for example by incorporating it into a game engine for rapid visualisation.

2.2. Data Representation

An appropriate data representation is necessary for interlocked processing and to enable recipe-like pipeline generation. In the *Habitat3D* framework each data entity is defined by the tuple $\mathbf{D} = \{\mathcal{C}, \mathcal{N}, \mathcal{I}, \mathcal{M}\}$. $\mathcal{C}$ is the actual point

cloud (either (x, y, z) or (x, y, z, r, g, b)). If normals N are extracted each point $p \in \mathcal{C}$ has an associated vector specifying the direction of the underlying surface ($|\mathcal{C}| = |\mathcal{N}|$). A cluster $c \in \mathcal{I}$ specifies a subset of points in $\mathcal{C}$ ($c \subset \mathcal{C}$ and $|\mathcal{I}| \leq |\mathcal{C}|$). All filtering steps can either be performed on the entire cloud $\mathcal{C}$ or the individual clusters $c \in \mathcal{I}$. The polygon mesh $\mathcal{M}$ unifies the vertices, edges and faces representing the polyhedral object.

2.3. Data Collection

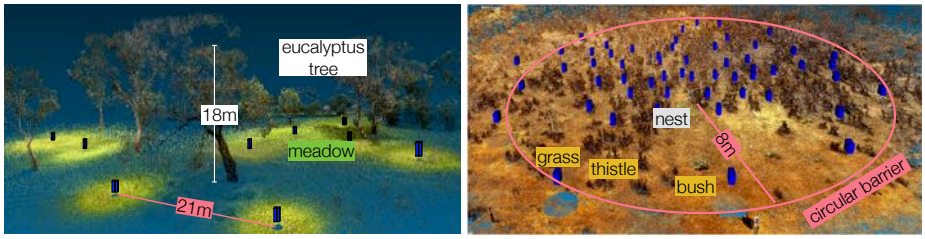
Two datasets were collected following the exact same protocol to quantify the performance of our system (called Canberra and Seville dataset). In order to test the flexibility and generalisability of Habitat3D we applied our system to two additional environments, namely a structured garden with trees, bushes, hedges and fences and a rough, volcanic surface.

2.3.1. Scanning

Colour point clouds were sampled using a laser scanner / colour camera combination (Z + F IMAGER 5006i, with an attached motorized colour camera (Z + F M-Cam), Zoller + Fröhlich GmbH, Wangen, Germany) as in [9]. After levelling the device, each scan followed a pre-programmed routine that firstly rotated the LiDAR and then the camera producing a high-resolution colour point cloud (angular resolution $10,000$ points/360° with a 360° horizontal and 310° vertical field of view, for a range of distances from 0.5 to 80m). Since we used a phase-based scanner both scanning and initial preprocessing might vary given time-of-light based scanners are used. The actual modelling procedure can be applied to scans from both technologies.

2.3.2. Registration

Several artificial markers (checkered pattern with two white and two black squares) were placed within the environments, providing easily identifiable reference points with which all scans were registered onto a common coordinate system. Using the accompanying Zoller + Fröhlich software (Z + F LaserControl) each of the markers is labelled in every scan allowing all scans to be



Canberra dataset Seville dataset

Figure 2: DATASET COLLECTION. Blue boxes indicate scanner positions. Different plants and several distance measures are indicated.

registered to a common frame of reference. The same markers are also identified in the set of camera images, which allows registration of the Z+F M-Cam with respect to the LiDAR coordinate system and mapping of colour from the RGB images onto the 3D points (again using Z+F LaserControl). In addition, Z+F LaserControl is used to remove erroneous range values, mainly in the sky region and at object boundaries (caused by the small but finite diameter of the laser beam, hitting two or more objects of different range), and to optionally subsample the cloud.

2.3.3. Canberra Dataset

In 2011, we captured 9 point clouds in an urban park in Canberra, Australia, where several ant colonies (species: *Myrmecia croslandi*) had their nests and participated in navigation experiments [26, 27]. The area covered about 8800m^2 featuring several large Eucalyptus trees exhibiting complex natural structures like sub-branches and leaf clusters (Figure 2). The 9 clouds in total comprise approximately 42 million points requiring 1.5Gb of memory. Distances between scanner positions varied from $\sim$ 20m to more than 100m and were placed to cope for the density of the vegetation.

2.3.4. Seville Dataset

The natural foraging environment of a single colony of desert ants (species: *Cataglyphis velox*) was mapped in the summer of 2012 on the outskirts of Seville, Spain. The experimental setting restricted ant foraging to a 1018m^2 area of

gritty desert containing more than $1,700$ plants (a mixture of grasses, thistles and other bushes). An artificial barrier was created for the behavioural study which circles the nest at the 8m radius and was 10cm in height. 56 laser scans were taken with the aim of viewing all plants from multiple locations limiting occlusions. The maximal distance between scans was $\sim$ 16m and the location of the scans are shown by the blue boxes in Figure 2. The positions were chosen to surround the ant nest and distributed to cope for the density of the vegetation. Following the registration and sub-sampling steps (c.f. Section 2.3.2) each of the 56 clouds $\mathcal{C}_i$ $(i = 1, ..., 56)$ contains N_i 3D RGB-D[1] points $p_i^j \in \mathcal{C}_i$ $(j = 1, ..., N_i)$ with $p_i^j = (x, y, z, r, g, b)$ and $3,231,214 \leq N_i \leq 3,673,825$. The total number of RGB-D points was $\sum_i N_i = 192,223,621$ requiring 6.95Gb of memory.

2.4. Meshing Pipeline

As illustrated in Figure 3 the pipeline to reconstruct these natural environments can be separated into three consecutive steps. First, the clouds are segmented into ground and vegetation points. Second the ground and vegetation clouds are processed and meshed separately to create the ground and vegetation mesh. Finally the two meshes are merged into the global model.

2.4.1. Semantic Segmentation

After loading all clouds $\mathcal{C}_i$ $(i = 1, ..., N)$ the first step of the pipeline is to segment these clouds into semantic subunits. To decrease the computational time, we separated all clouds into chunk clusters containing $10,000$ RGB-D points at maximum (see Appendix A.3.3 *Iterative chunk split* for details). Then, progressive morphological filtering was performed on each cluster separately to extract ground clouds $\mathcal{G}_i$ and vegetation clouds $\mathcal{V}_i$. This filter was initially introduced to remove non-ground measurements from airborne LiDAR [28] and utilises the difference in elevation and an opening operation within a custom sized window. In order to apply this filter to terrestrial scans we used its high specificity in

[1] RGB-D is an abbreviation for colour (r,g,b) and position + depth (x,y,z) information.

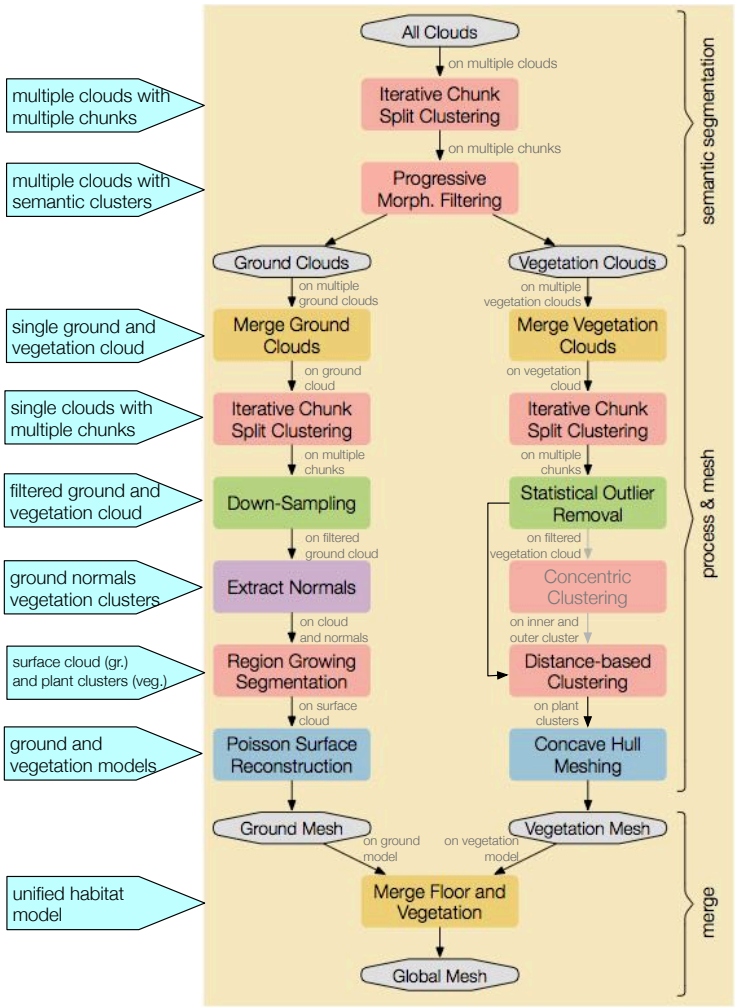


Figure 3: Meshing pipeline overview. Grey boxes represent data structures and yellow, green, red, violet and blue boxes indicate general processing, point cloud filtering, clustering / segmentation, feature extraction and meshing respectively. Box with light grey text indicates an optional processing step. Resultant clouds / cloud clusters are given in the bright blue boxes on the left. Inputs to each processing step are given next to the arrows.

order to extract the vegetation points $\mathcal{V}$ by tuning the parameters to ensure very close-to-the-ground vegetation segmentation (for details see Appendix A.3.3). This filter was chosen because no preliminary filtering is necessary to identify even very subtle vegetation structures (whereas filtering could interfere with

the topology of these structures). The resultant ground and vegetation clouds are subsequently processed independently. Since this filter identifies the ground based on abrupt elevation differences, non-ground points, which form a regular surface, might still be included in $\mathcal{G}_i$. This is addressed in a subsequent segmentation step to ensure points on the main surface only (c.f. Section 2.4.2).

2.4.2. Process & Mesh

To extract meshes for both the ground and the vegetation, different reconstruction strategies are necessary in order to model natural outdoor scenes, since both provide particular challenges. The requirements for ground meshing are:

- finding a controllable balance between resolution and smoothness
- being sufficiently resilient to noise and gaps and
- taking the properties of the former ground segmentation into account (i.e. smooth surface points with low overall curvature)
- ensuring a watertight ground model to avoid gaps and to support game engine physics (e.g. collision control)

The requirements for meshing different types of plants are:

- being tolerant to irregular distributions of normals and thus highly dynamic surface geometries present in naturally grown vegetation
- enabling meshing on non-optimised outer boundaries to preserve most of the natural structures (filtering techniques can blur the appearance since they rely on certain underlying geometrical assumptions)
- being tolerant to incorrect plant point classifications (i.e. a single plant is segmented into several sub-clusters or a single cluster contains multiple plants or even non-plant points) and
- being able to control the distance that defines if neighbouring points are connected

The latter requirement implies finding a balance between smoothing across contiguous surfaces without merging nearby but distinct structures (e.g. nearby flowers; see also Figure 8 for an example).

Ground Meshing. After merging all ground point clouds $\bigcup_{i=1}^{N} \mathcal{G}_i = \mathcal{G}$, another chunk split clustering is mandatory to further process the cloud since the resultant ground cloud can cover a huge area and might contain millions of redundant points (cf. Table 1). All subsequent steps (besides meshing) are facilitated on the resultant clusters. To further address this redundancy, we applied downsampling to regularly distribute the points on a grid and extracted the normals for each ground point (relative to the global coordinate system origin).

As described in Section 2.4.1, progressive morphological filtering was used to extract the ground and vegetation based on abrupt jumps in elevation. Since this filter *removes* points above the approximated underlying surface, points below this surface are not explicitly removed. Furthermore, progressive morphological filtering does not ensure a single smooth surface with low curvature. Therefore, we applied region growing segmentation to $\mathcal{G}$, which allows us to arbitrarily control smoothness and curvature (for details see Appendix A.3.3). Finally, Poisson surface reconstruction was used to extract the ground mesh which accounts for all above listed requirements and guarantees a watertight model. The resolution of the surface can be controlled by the depth of the underlying adaptive octree (high depths result in capturing finer details, low depths result in a smoother surface; for details see Appendix A.4). Furthermore, this strategy is very resilient to noise and gaps since all ground points $\mathcal{G}$ are meshed at once (for details see Appendix A.3.4). Note that both region growing segmentation and Poisson surface reconstruction are based on the surface normals and complement one another so that the underlying geometry can be preserved (Section 3).

Vegetation Meshing. Equivalent to the ground processing the vegetation clouds are merged into a global vegetation cloud $\bigcup_{i=1}^{N} \mathcal{V}_i = \mathcal{V}$ and separated into chunk clusters. This time however, the chunk clusters were only used to speed up the

statistical outlier removal. As a result, the plants in $\mathcal{V}$ are much sharper since noisy points and registration artefacts are removed.

The next step is to identify individual plants based on their mutual distance and compactness (see Section 2.1). However, depending on the size (height) of the plants, the laser scanner resolution and the distance to the scanner, the point density per plant can vary drastically. Therefore, the merged cloud can be clustered into concentric regions around the (main) scanner position optionally (see Figure 3) so that plant clustering can be done on these regions separately. If concentric clustering was used the cluster tolerance of this distance-based plant clustering has to be adjusted accordingly (low for high compactness and high for low compactness). Otherwise distance-based clustering with one global cluster tolerance is applied to $\mathcal{V}$ (for a detailed discussion of the parameters see Appendix A.4).

Laser scanners strobe the outer boundary of the objects within the scanning radius but inaccuracies during scanning and merging can cause points *inside* these boundaries. To extract the outer boundary of the underlying plant cluster volumes concave hulls (also called alpha-shapes) can be used which are general enough to be applicable for different types of plants and independent to the surface normals. Over-fitting can be avoided by a parameter α which limits the size of the resultant hull segment (for details see Appendix A.3.4). Since this parameter can be used to control which neighbouring points are connected, α also relies on the underlying compactness and has to be chosen according to cluster tolerance. Thus, concave hull meshing fullfills all vegetation reconstruction requirements listed above. If concentric clusters have been extracted, α should increase in proportion to the increased cluster tolerance.

2.4.3. Merge Meshes

Both Poisson surface reconstruction and concave hull meshing can generate arbitrarily detailed polygon meshes. Depending on the application, the meshes can be directly merged or retopologised by applying mesh filters as described in Appendix A.3.5. The resultant models can be concatenated since they still

share the same global vertex coordinate system. It should be noted that a small gap can appear between the ground surface (especially if the ground model has a low octree depth) and concave hull plants which can be removed by retopologisation. In either case, the resultant mesh, the point clouds as well as the normals and clusters are exported in several well-known formats, namely *ply*, *obj*, *vtk* and *pcd*.

3. Results

We tested our *Habitat3D* framework on two distinct datasets, one taken in Canberra, Australia and the other one sampled in Seville, Spain. In both cases the scanning was aimed at mapping the foraging habitat around an ant nest. The Canberra dataset features a comparatively large meadow area including several large Eucalyptus trees, whereas the Seville dataset comprises a densely vegetated desert scene with more than thousand smaller plants such as thistles. The two datasets strongly differ in the amount of scans: Canberra is sparsely imaged using only 9 scans whereas 56 scans were made within a 200m^2 circular area in Seville. The resultant reconstructions of both datasets as well as the intermediate results are evaluated qualitatively and quantitatively in this section. We also evaluated two additional datasets to demonstrate the generalisability of Habitat3D (Section 3.3). All calculations, evaluations and visualisations are done on an Intel Xeon E3-1245 3.4GHz computer equipped with 32 Gb DDR3 RAM and an NVIDIA Quadro K4200 graphics card.

3.1. Canberra Model

After progressive morphological filtering the ground ($\mathcal{G}$) and vegetation points ($\mathcal{V}$) are successfully separated. Since we set the filter to operate with high specificity we ensure a very close-to-the-ground vegetation segmentation leading to almost no false-positive points in the ground cloud, but some irregular points close to the ground appear in the vegetation cloud (c.f. symbol '$*$' in Figure 4 *semantic segmentation*). The overall processing time for all 9 clouds is $13,097$s,

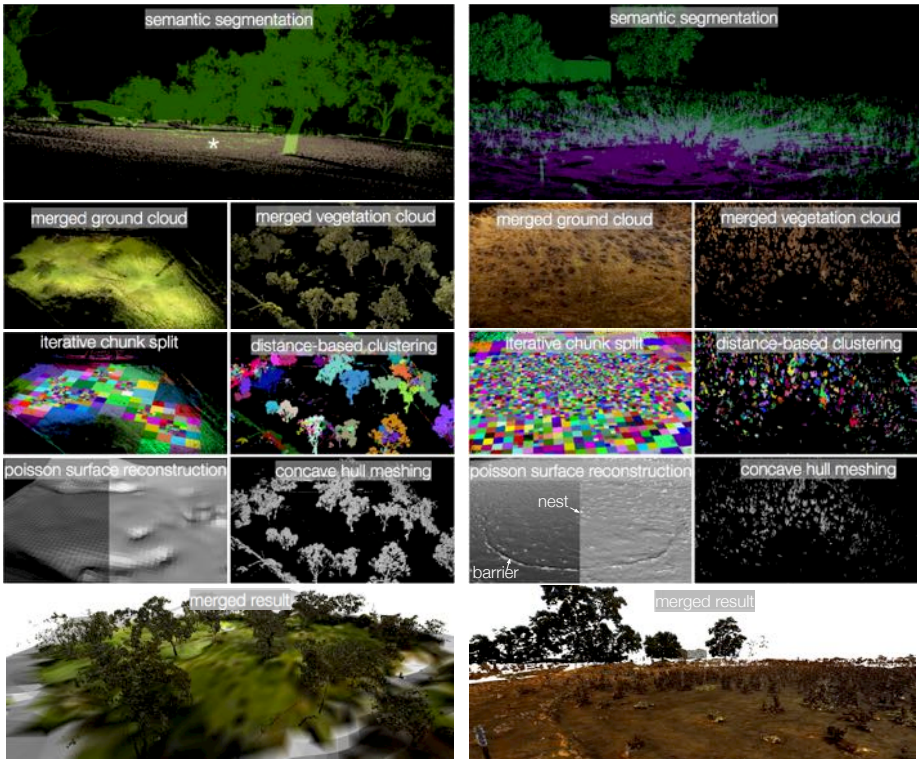


Figure 4: NATURAL ENVIRONMENT RECONSTRUCTION STEPS. Text at the top of each sub-figure corresponds to the respective processing steps given in Figure 3. For details see text.

thus, semantic segmentation of a single cloud takes about 24 minutes, which is by far the most time consuming step of the pipeline (c.f. Table 1).

Figure 4 *iterative chunk split* shows the result of recursively splitting the ground into chunks containing $100,000$ points at maximum. Note that the different scanning positions can be identified based on the increased point density around the terrestrial LiDAR scanners. After down-sampling, normal extraction and region growing segmentation the ground meshes are extracted by using Poisson surface reconstruction. To demonstrate how meshes can be used to extract compact representations of huge areas we intentionally used a low octree depth of 6 (c.f. Figure 4 *poisson surface reconstruction*) resulting in a mesh made up of $3,564$ polygons. After colourisation, the ground model needs only

Figure 5: DIFFERENT PLANT RECONSTRUCTIONS. Segmented point clouds are given on the left and resultant meshes are given on the right. Close up meshes are given as wired surfaces. The dimension is indicated.

276Kb disk storage, which is 0.024% of the memory used by the original ground clouds $(1,138,712\text{Kb})$.

For individual plant identification in $\mathcal{V}$ distance-based clustering is used. As visible in Figure 4 *distance-based clustering* densely sampled trees in the middle of the scene are correctly clustered into a single object, whereas distant trees are fragmented. Since concave hull meshing neither requires continuous nor correctly segmented plants, meshing the vegetation is not affected by these subdivisions (c.f. Figure 4 *concave hull meshing*) and the small cluster sizes are advantageous in terms of computational time (only $1,037$ seconds or 17 minutes are necessary to reconstruct all vegetation). In contrast to the ground model, the vegetation model is not reduced in size, as almost all points in $\mathcal{V}$ are used to preserve the complex topology by using a small α value to build the alpha shapes (c.f. Appendix A.3.4; mesh filtering can be used if lower vegetation reconstructions are required as explained in Appendix A.3.5). After colourisation, the vegetation model needs 470Mb of memory which is slightly higher than the amount of memory used for $\mathcal{V}$ (requiring 385Mb).

A tree scanned from all directions is shown in Figure 5. Point clouds are given on the left and resultant reconstructions are given on the right. Note that all natural structures up to the very thin ramification are reconstructed correctly (obviously where branches are obscured by leaves they cannot be reconstructed). The reconstructed tree is approximately 20m high. Even though terrestrial scanning was used, a realistic reconstruction was possible up to the very top of the plant. Finally the resultant ground and vegetation models are colourised and merged to generate the overall Canberra model shown in Figure 4 *merged results*.

The accuracy of the used LiDAR hardware for natural environments has already been demonstrated elsewhere [9]. To evaluate the accuracy of our meshing pipeline we calculated the mean distance and standard deviation between the raw clouds and the resultant mesh using a third party software called CloudCompare. Since the 9 clouds cover a comparatively large area, we only use a circular area with 20m radius covered by all clouds (i.e. area of interest). Mean and standard deviation for each cloud is given in Figure 6 top and the overall accuracy of all clouds is given in Figure 6 bottom. The median distance between points of all clouds and the resultant mesh is below 6mm and the highest measured mean distance is 3.3cm. Since the ground mesh is intentionally meshed with low resolution and terrestrial scans are used to mesh large trees the resultant millimetre accuracy clearly demonstrates the overall high reconstruction precision. Furthermore, the raw point clouds contain noise (random points in the sky) which are successfully removed using our pipeline but increase the mean deviation in this evaluation.

3.2. Seville Model

As described in the Section 2.3.4, this dataset consisted of a much higher number of scans within a smaller area, enabling more accurate reconstructions. Since the 1018m^2 ground cloud $\mathcal{G}$ comprises $156,174,403$ points iterative chunk splitting (maximal $100,000$ points) results in $3,383$ clusters, again with higher density around the terrestrial scanner positions (c.f. Figure 4 *iterative chunk*

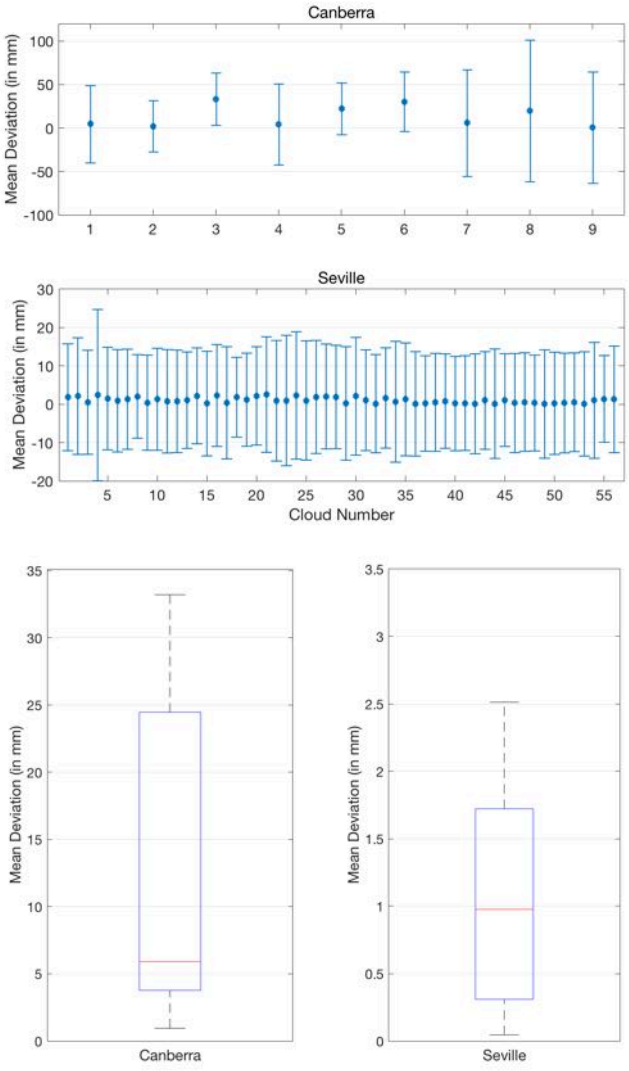


Figure 6: Quantitative Accuracy Evaluation. The resultant meshes are compared with all raw clouds. Left: mean deviation and standard deviation between each cloud and the mesh in mm. Right: mean deviations summaries in box plots for both datasets.

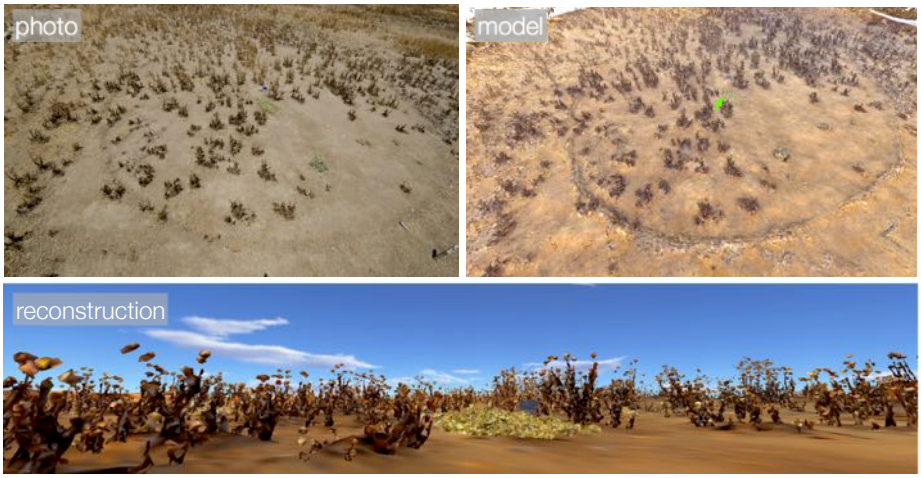


Figure 7: Natural environment reconstruction results. Top: Photography vs. meshed model of the Seville dataset. Bottom: Rendered reconstruction from an ant's perspective in the Seville dataset (position indicated by a green arrow).

split). In contrast to the Canberra dataset, subsequent Poisson surface reconstruction was tuned towards high accuracy by using an octree depth of 10. The result is 897, 106 polygons requiring 120Mb memory but still preserving all main topological features. For example the artificial barrier is reconstructed with very high accuracy as shown in Figure 4 *poisson surface reconstruction* and the undulating surface structure (mainly caused by accumulated depositions below the plants) as well as the small hill surrounding the nest entrance are well preserved (cf. Figure 7).

Euclidean distance based clustering of $\mathcal{V}$ results in 3, 539 individual plant clusters (c.f. Figure 4 *distance-based clustering*). Since the maximal allowed cluster size was set to 50 points, small fragments like branches, leafs and other irregularities close to the ground cause this oversized number of clusters. However, latter concave hull meshing is not affected by these fragmentations resulting in correctly meshed vegetation as shown in Figure 4 *concave hull meshing*.

Examples of different plant types are given in Figure 5. For the thistle, it can be seen that concave hull meshing can preserve small holes and mesh tiny structures such as thorns. However, if the gap between the flower and the stem

is too large, flowers can appear disconnected from their stems, i.e. free-floating. This could only be avoided by increasing α (cf. Appendix A.3.4), which would also merge disjoint structures. An example of incorrectly merged structures like flowers, mainly caused by close proximity and the non-rigid character of the plants, can be seen in Figure 5 (e.g. highest flowers of the left thistle). These artefacts could potentially be addressed by game engine specific modifiers to adjust the mesh. The points of an example cluster of withered grass can be seen in Figure 5 *grass*. Also using concave hull meshing, with $\alpha = 1$cm the branching structure at the top of the grass is preserved in high detail (cf. Figure 5 close-up). A low-to-the-ground green bush is shown in Figure 5 *bush*. Due to the flat appearance of the bushes, either triangle meshing or concave hull meshing can be used to reconstruct this plant. However, both strategies fail to generate a smooth surface so that the meshes result in sharp-edged and grainy polygons. Despite this, the overall topology of the clouds is well preserved.

The merged result given in Figure 4 indicates the accuracy of overall meshing (distant objects outside the 1018m^2 radius are also meshed and added to the model). Figure 7 provides a comparison between a real photo and the artificial reconstruction rendered from the equivalent location. Note that colour gradients in the ground texture are kept in the reconstruction due to the vertex attribute transfer. The resultant model only needs only 1.5% of the initial memory requirements of the original point clouds (c.f. Table 2).

In contrast to the Canberra dataset 56 clouds were used to densely sample a region of 1018m^2. In Figure 6 the deviations between all raw clouds within this region and the resultant mesh are given. The median distance is below 1mm and the maximum measured mean distance is 2.5mm. As shown in Figure 6 top only cloud number 4 has a standard deviation above 2cm. Thus, the resultant deviations are mainly caused by noise (random points in the sky) and points inside the reconstructed surfaces, mainly caused by registration errors and moving objects. This illustrates that Habitat3D can be used to reconstruct 3D meshes from point clouds with sub-millimetre precision.

3.3. Other Models

To demonstrate the generalisability we used Habitat3D to extract models of two additional environments, namely a well-structured garden captured in Wageningen (Netherlands) and a volcano surface scanned on Mount Etna (Italy). The garden dataset consists of $16,265,804$ points and features 3 trees, 19 bushes, fences, two paths and several artificial objects in the back (car, container, etc.)[2]. The model was generated using the same pipeline and parameters as used for the Seville dataset. As shown in Figure 8 (top) the ground and vegetation are correctly reconstructed. An overgrown fence is additionally given in Figure 8 top right: Habitat3D correctly preserves most of the holes while still providing the overall structure of the fence. Partial vegetation on the fence can also be seen. However, some wires are missing leading to holes in the grid.

The volcano dataset does not feature any vegetation and consists of 6 point clouds ($8,721,951$ points) showing a rough, volcanic surface on Mt Etna. The scans were recorded within the ROBEX project[3] for simulation purposes in preparation of the ROBEX moon analogue mission [29, 30]. The model shown in Figure 8 (bottom) was generated using only the ground meshing pipeline with the same parameter settings as in the Seville dataset. As shown in the bottom right the overall shape of the ground is correctly reconstructed with high precision. However, the edgy shape of some stones appears unrealistically smooth due to the use of Poisson surfaces. The size of the dataset reduces from 338.4 MB to 40.8 MB (reduction of 88%). Furthermore, the surface shown in Figure 8 (bottom right) demonstrates how specialised shaders can be applied to improve the analysis of LiDAR data: the surface was rendered using the electronic microscope shader which visually emphasises the overall three dimensional structure.

[2] Torsten Sattler, Thomas Brox, Marc Pollefeys, Robert B. Fisher, Radim Tylecek. 3D Reconstruction meets Semantics Reconstruction Challenge, ICCV Workshops, October 2017. URL: http://trimbot2020.webhosting.rug.nl/events/3drms/challenge/

[3] Project website www.robex-allianz.de/en/

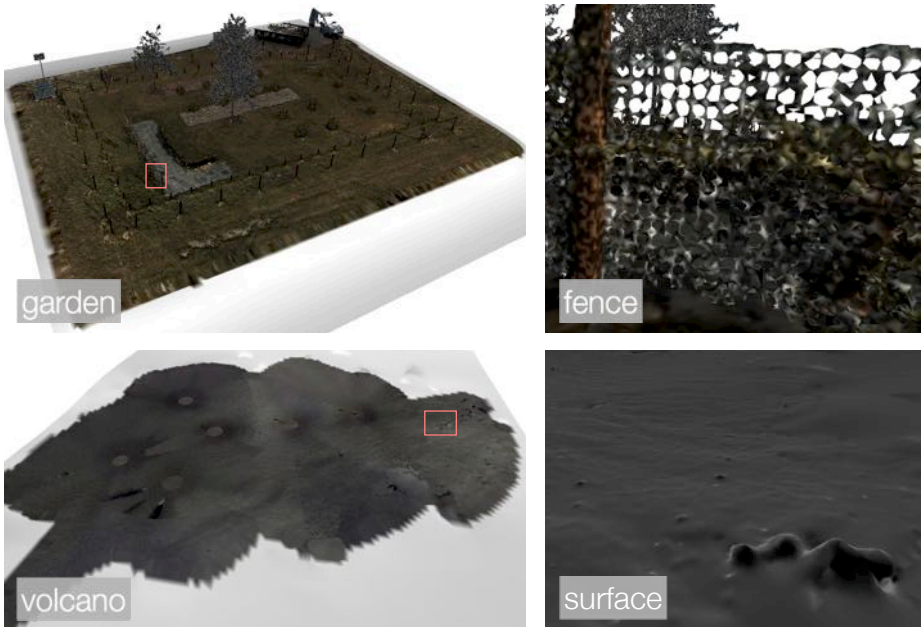


Figure 8: Reconstructions of additional environments. Top: Garden dataset featuring several trees, bushes, man-made paths and a fence (top right for a close up indicated by the red box). Bottom: Volcano dataset from a test site of the ROBEX analogue mission campaign on Mount Etna. Close-up showing the surface is given in the bottom right.

4. Discussion & Conclusion

We have introduced Habitat3D: a novel, cross-platform, open-source and generic software framework to extract precise models of natural environments that have been mapped using a 3D point-cloud producing technology such as LiDAR. The system bridges the current gap between defacto environmental mapping hardware (laser scanners) and similarly mature software and hardware systems (game engines and associated graphics cards) allowing for rapid visualisation, manipulation and analysis of complex scenes. The framework produces meshes, rather than digital elevation models (DEM) or extracting triangulated irregular networks (TIN) and accurately recovers the underlying surfaces in the natural environment. Since segmentation is required for our framework, individual semantic subunits like ground or individual plants can be processed and

analysed independently.

Using meshes has several advantages compared to a usage of point clouds directly. For example, meshes are highly compressed, and are in a format that can leverage state-of-the-art hard- and software such as graphics cards and game engines for improved visualisation and analysis. The memory usage and processing times are summarised in Tables 1 and 2. The reduction of memory usage is illustrated in Figure 9 indicating very high efficiency in compressing the registered and merged clouds (which usually overlap to cover larger areas causing high oversampling). However, compressing the data using meshes inevitably removes information which can affect the quality of the final model. Since specialised shaders and mapping strategies can be applied to meshes this loss in information can be partially compensated. For example, bump mapping can be used to simulate a more complex surface on simplified polygon meshes [31]. Furthermore, meshes comprise a topology and appear as continuous objects allowing accurate visualisation even in close proximity and implicitly contain a geometry (e.g. volume extraction). Finally, lightning and shadows can be added, collision control can be implemented and physics can be incorporated.

Reconstructing natural scenes using ground-based LiDAR requires the extraction of fundamental properties. For example, considering the overall geometry no filtering should be applied before ground / vegetation segmentation since different processing strategies are required: the ground points should form a wavy sheet whereas plant points should define a volume after processing. Furthermore, geometrical properties need to be identified to extract semantic sub-units from raw clouds (c.f. Figure 1). An example of the technical properties is that the point density decreases quadratically with the distance to a ground-based scanner and inaccuracies of scanning itself have to be addressed. Thus, processing subsets or changing filtering parameters relative to the LiDAR scanner distance improves both computational time and overall results.

However, challenges remain when working with and converting to meshes. For example, for distal objects, clouds might look qualitatively better compared to opaque meshes. This can, for example, be addressed by introducing trans-

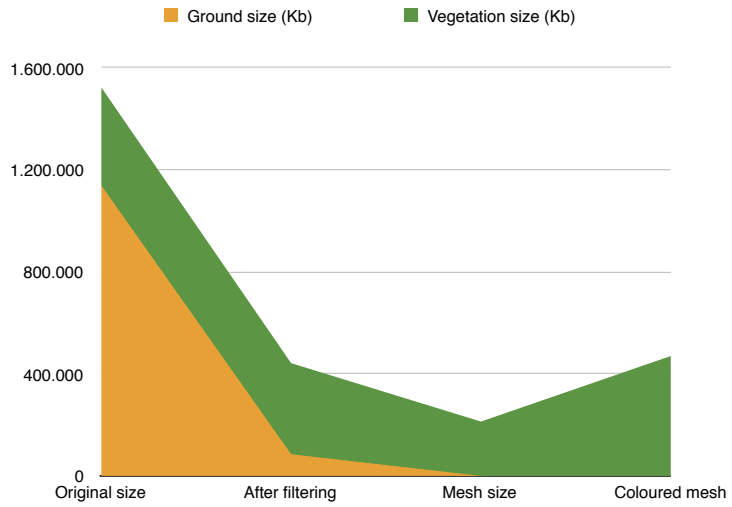
parency (alpha blending) which is a standard technique in Computer Graphics and again optimised for meshes. In addition, as objects can be misaligned across scans, and thus appear bloated in the final concatenated cloud, a more elaborate merging algorithm could be used to reduce oversampling and redundancy. Habitat3D provides different meshing and mesh processing strategies but only guarantees watertight surfaces (which are necessary for volume extraction) by using Poisson surface reconstruction. In case watertight surfaces are required Poisson surface reconstruction should be used to generate the meshes. Finally, meshing different natural outdoor scenes involves manual trial and error to determine the optimal routines and associated parameters for best modelling results. We addressed this issue by implementing a GUI and user feedback of all modelling steps and recipe-like batch processing. However, some form of data-driven parametrisation is desirable to reduce user workload and may increase accuracy.

In the future we intend to extend the framework by implementing a more selective cloud merging strategy to overcome the bloated appearance of several vegetation clusters. If reconstructed meshes of complex vegetation is desired, the most common method is to use skeletonisation approaches to extract the three-dimensional geometry of the plant. For example, directed graphs and weak constraints have been used to guide a global optimisation for tree-skeleton reconstruction [32]. Others have used the Dijkstra algorithm to extract the tree skeleton by clustering boughs based on their distance to the root point [17]. Normal- and L1-medial-based skeletonisation algorithms have also been used for complex point cloud topologies [33, 34]. Since the segmentation and filtering described above lead to very accurate and noise-free plant clusters, these sub-units are ideally suited for these approaches. Skeletonisation can also help to identify ghosting artefacts like branches moving in the wind. Thus, we plan to add skeletonisation algorithms to improve the appearance of natural structures like trees and to introduce a high-level topology of plants (e.g. to make them move with wind etc.) in newer versions of Habitat3D.

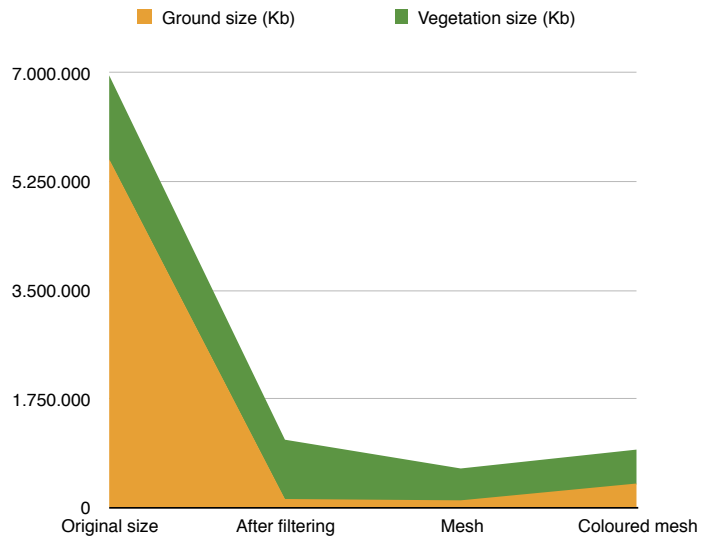
The specific motivation for the development of the *Habitat3D* system was

Table 1: Computational complexity of the Canberra dataset. Size, memory usage (Kb) and computational time (ms) of modelling the Seville dataset. The size dimension is given in column Dim. (pts = points, clu = clusters, nor = normals and pol = polygons). We use the memory usage of uncompressed non-binary ply files to measure the used Kb. Processing steps correspond to Figure 3.

Step	Size	Dim.	Memory	Time
All clouds (9)	41,989,885	pts	1,523,356	-
Exmpl. cloud	4,883,777	pts	169,647	-
Prog. Morph. (9)	41,989,885	pts	1,523,356	13,096,978
Merged Ground	31,170,464	pts	1,138,712	-
Chunk Split	816	clu	269,426	197,341
Down-Sampl.	2,325,996	pts	91,144	35,971
Normal Ex.	2,325,996	nor	37,220	6,331
Region Grow.	2,165,870	pts	85,026	25,035
Poisson Surf.	3,564	pol	106	3,539
Coloured Ground	3,564	pol	276	-
Merged Veget.	10,820,521	pts	384,644	-
St. Out. Rem.	10,048,165	pts	357,115	299,047
Dist. Cluster.	1,400	clu	79,952	514,768
Concave Hull	12,667,922	pol	213,012	1,037,440
Coloured Veget.	12,667,922	pol	469,833	-
Merged Model	12,671,486	pol	213,118	-
Colour Model	12,671,486	pol	470,109	-



Canberra dataset



Seville dataset

Figure 9: Changes in memory usage. Changes in size (Kb) of the ground (yellow) and vegetation (green) processing. We use the memory usage of uncompressed non-binary ply files to measure the used Kb. After filtering includes all step before meshing and mesh and coloured mesh indicate the memory needed for the respective polygons after meshing.

Table 2: Computational complexity of the Seville dataset. Size, memory usage (Kb) and computational time (ms) of modelling the Seville dataset. The size dimension is given in column Dim. (pts = points, clu = clusters, nor = normals and pol = polygons). We use the memory usage of uncompressed non-binary ply files to measure the used Kb. Processing steps correspond to Figure 3.

Step	Size	Dim.	Memory	Time
All clouds (56)	192,223,621	pts	6,950,754	-
Exmpl. cloud	3,603,707	pts	126,184	-
Prog. Morph (56)	192,223,621	pts	6,950,754	58,766,296
Merged Ground	156,174,403	pts	5,613,847	-
Chunk Split	3,383	clu	1,450,636	3,722,901
Down-Sampl.	6,581,853	pts	258,856	431,736
Normal Ex.	6,581,853	nor	105,314	19,747
Region Grow.	4,657,151	pts	184,960	61,174
Concen. Clu.	3,587,668	pts	141,734	643
Poisson Surf.	897,106	pol	120,199	84,546
Coloured Ground	897,106	pol	391,067	-
Merged Veget.	36,050,173	pts	1,348,755	-
St. Out. Rem.	34,021,556	pts	1,273,780	1,149,657
Concen. Clu.	25,167,697	pts	954,728	4,375
Dist. Cluster.	3,539	clu	289,687	2,747,506
Concave Hull	29,339,855	pol	513,090	2,369,191
Coloured Veget.	29,339,855	pol	546.008	-
Merged Model	30,236,961	pol	633,289	-
Colour Model	30,236,961	pol	937,075	-

to map the natural habitat of ants, in order to facilitate the evaluation of hypotheses regarding their sensorimotor and navigation strategies. For example, Figure 7 shows a rendered image of the Seville habitat from an insect's perspective. Note that the mesh was down-sampled quadratically to increase the rendering performance, which is crucial for testing algorithms of visual navigation. Even after down-sampling, all characteristics remain visible in the rendered

views. Furthermore, a sky-dome and (sun) light source was added to improve realism, and lightning and sky patterns can be altered to represent different times of day. More generally, to understand animal behaviour it is important to characterise the natural environments in which they live and behave [35] and the approach we have presented can make a significant contribution to that aim. However we believe the framework we have developed can be useful in a variety of applications, including biomass measurements and agriculture [6, 11], forestry [36, 37, 38], ecology and conservation [39], flood modelling [40, 41], archaeology [23, 42], geology [5], building surveys [43], virtual reality [44], mobile robotics and game development. Other applications require other pipelines and strategies of course. If for example individual leaves need to be reconstructed more fine grain clustering has to be applied to the plant clusters. Given very dense vegetation (e.g. rain forest canopies) additional separation routines are required to identify individual plants. Since our focus was photorealistic rendering of natural outdoor scenes we do not exhaustively address merged vegetation in our pipeline.

Both the meshing framework (*Habitat3D*) and the resultant 3D world models of the animal habitats are available as open source downloads at the Ant Navigation Challenge website.

Acknowledgement

The authors would like to thank Antoine Wystrach for helping with the data collection in Seville. We would also like to thank the TrimBot2020 team (in particular Bob Fisher and Radim Tylecek) for providing the garden dataset and Thomas Abmayr (Munich University of Applied Sciences) for providing the point clouds of the volcano data set. This project is funded by the BBSRC and the EPSRC (EP/M008479/1). We would like to acknowledge the support of NVIDIA Corporation with the donation of graphics hardware used for this project.

References

[1] M. A. Lefsky, W. B. Cohen, G. G. Parker, D. J. Harding, Lidar Remote Sensing for Ecosystem Studies, BioScience 52 (1) (2002) 19–30.

[2] K. Lim, P. Treitz, M. Wulder, B. St-Onge, M. Flood, LiDAR remote sensing of forest structure, Progress in physical geography 27 (1) (2003) 88–106.

[3] J. C. Vogeler, W. B. Cohen, A review of the role of active remote sensing and data fusion for characterizing forest in wildlife habitat models, Revista de Teledetección (45) (2016) 1–14.

[4] G. J. Newnham, J. D. Armston, K. Calders, M. I. Disney, J. L. Lovell, C. B. Schaaf, A. H. Strahler, F. M. Danson, Terrestrial Laser Scanning for Plot-Scale Forest Measurement, Current Forestry Reports 1 (4) (2015) 239–251.

[5] A. T. Hudak, J. S. Evans, A. M. Stuart Smith, LiDAR Utility for Natural Resource Managers, Remote Sensing 1 (4) (2009) 934–951.

[6] P. Michel, J. Jenkins, N. Mason, K. J. M. Dickinson, I. G. Jamieson, Assessing the ecological application of lasergrammetric techniques to measure fine-scale vegetation structure, Ecological Informatics 3 (4-4) (2008) 1–12.

[7] A. B. Davies, C. J. Tambling, G. I. H. Kerley, G. P. Asner, Effects of Vegetation Structure on the Location of Lion Kill Sites in African Thicket, PloS ONE 11 (2) (2016) e0149098–20.

[8] X. Yang, C. Schaaf, A. Strahler, T. Kunz, N. Fuller, M. Betke, Z. Wu, Z. Wang, D. Theriault, D. Culvenor, D. Jupp, G. Newnham, J. Lovell, Study of bat flight behavior by combining thermal image analysis with a LiDAR forest reconstruction, Canadian Journal of Remote Sensing 39 (2014) 112–125.

[9] W. Stürzl, I. Grixa, E. Mair, A. Narendra, J. Zeil, Three-dimensional models of natural environments and the mapping of navigational information., Journal of Comparative Physiology A 201 (6) (2015) 563–584.

[10] W. Stürzl, J. Zeil, N. Boeddeker, J. M. Hemmi, How Wasps Acquire and Use Views for Homing, Current Biology 26 (4) (2016) 470–482.

[11] M. Dassot, A. Colin, P. Santenoise, M. Fournier, T. Constant, Terrestrial laser scanning for measuring the solid wood volume, including branches, of adult standing trees in the forest environment, Computers and Electronics in Agriculture 89 (2012) 86–93.

[12] A. S. Antonarakis, K. S. Richards, J. Brasington, M. Bithell, Leafless roughness of complex tree morphology using terrestrial lidar, Water resources research 45 (10).

[13] H. G. Maas, The suitability of airborne laser scanner data for automatic 3D object reconstruction, Ascona01 (2001) 291–296.

[14] C. Früh, An Automated Method for Large-Scale, Ground-Based City Model Acquisition, International Journal of Computer Vision 60 (1) (2004) 5–24.

[15] B. Gorte, N. Pfeifer, Structuring laser-scanned trees using 3D mathematical morphology, International Archives of Photogrammetry and Remote Sensing 35 (B5) (2004) 929–933.

[16] N. Pfeifer, B. Gorte, D. Winterhalder, Automatic reconstruction of single trees from terrestrial laser scanner data, in: Proceedings of 20th ISPRS Congress, 2004, pp. 114–119.

[17] H. Xu, N. Gossett, B. Chen, Knowledge and heuristic-based modeling of laser-scanned trees, ACM Transactions on Graphics 26 (4) (2007) 19.

[18] J.-F. Côté, R. A. Fournier, G. W. Frazer, K. O. Niemann, A fine-scale architectural model of trees to enhance lidar-derived measurements of forest canopy structure, Agricultural and forest meteorology 166 (2012) 72–85.

[19] W. Kazmi, S. Foix, G. Alenyà, H. J. Andersen, Indoor and outdoor depth imaging of leaves with time-of-flight and stereo vision sensors: Analysis and comparison, ISPRS journal of photogrammetry and remote sensing 88 (2014) 128–146.

[20] S. Paulus, H. Schumann, H. Kuhlmann, J. Léon, High-precision laser scanning system for capturing 3D plant architecture and analysing growth of cereal plants, Biosystems Engineering 121 (2014) 1–11.

[21] P. Raumonen, M. Kaasalainen, M. Åkerblom, S. Kaasalainen, H. Kaartinen, M. Vastaranta, M. Holopainen, M. Disney, P. Lewis, Fast automatic precision

tree models from terrestrial laser scanner data, Remote Sensing 5 (2) (2013) 491–520.

[22] J. Hackenberg, M. Wassenberg, H. Spiecker, D. Sun, Non destructive method for biomass prediction combining tls derived tree volume and wood density, Forests 6 (4) (2015) 1274–1300.

[23] G. Papaioannou, E.-A. Karabassi, T. Theoharis, Virtual Archaeologist: assembling the past, Computer Graphics and Applications, IEEE 21 (2) (2001) 53–59.

[24] K. M. Wurm, A. Hornung, OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems, Proc. of the ICRA workshop on best practice in 3D perception and modeling for mobile manipulation 2.

[25] N. Brodu, D. Lague, 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology, ISPRS Journal of Photogrammetry and Remote Sensing 68 (2012) 121–134.

[26] A. Narendra, S. Gourmaud, J. Zeil, Mapping the navigational knowledge of individually foraging ants, Myrmecia croslandi., Proceedings of the Royal Society of London B: Biological Sciences 280 (1765) (2013) 20130683.

[27] J. Zeil, A. Narendra, W. Stürzl, Looking and homing: how displaced ants decide where to go, Philosophical Transactions of the Royal Society of London B: Biological Sciences 369 (1636) (2014) 20130034.

[28] K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, C. Zhang, A progressive morphological filter for removing nonground measurements from airborne LIDAR data, IEEE Transactions on Geoscience and Remote Sensing 41 (4) (2003) 872–882.

[29] A. Wedler, M. Hellerer, B. Rebele, H. Gmeiner, ROBEX: Components and methods for the planetary exploration demonstration mission, 2015.

[30] A. Wedler, M. Vayugundla, H. Lehner, P. Lehner, M. J. Schuster, S. G. Brunner, W. Stuerzl, A. Dmel, H. Gmeiner, B. Vodermayer, B. Rebele, I. Grixa, K. Bussmann, J. Reill, B. Willberg, A. Maier, P. Meusel, F. Steidle, M. Smisek, M. Hellerer, M. Knapmeyer, F. Sohl, A. Heffels, W. L, C. Lange, R. Rosta,

N. Toth, S. Voelk, A. Kimpe, P. Kyr, M. Wilde, First results of the ROBEX analog mission campaign: robotic deployment of seismic networks for future lunar missions, 2017.

[31] L. Wang, X. Wang, X. Tong, S. Lin, S.-M. Hu, B. Guo, H.-Y. Shum, View-dependent displacement mapping., ACM Transactions on Graphics 22 (3) (2003) 334.

[32] Y. Livny, F. Yan, M. Olson, B. Chen, H. Z. 0002, J. El-Sana, Automatic reconstruction of tree skeletal structures from point clouds., ACM Transactions on Graphics 29 (6) (2010) 151.

[33] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, B. Chen, L1-medial skeleton of point cloud, ACM Transactions on Graphics 32 (4).

[34] A. Tagliasacchi, H. Zhang, D. Cohen-Or, Curve skeleton extraction from incomplete point cloud, ACM Transactions on Graphics 29 (3).

[35] K. T. Vierling, L. A. Vierling, W. A. Gould, S. Martinuzzi, R. M. Clawges, Lidar: shedding new light on habitat characterization and modeling, Frontiers in Ecology and the Environment 6 (2) (2008) 90–98.

[36] F. Morsdorf, C. Nichol, T. Malthus, I. H. Woodhouse, Assessing forest structural and physiological information content of multi-spectral LiDAR waveforms by radiative transfer modelling, Remote Sensing of Environment 113 (10) (2009) 2152–2163.

[37] M. Béland, J.-L. Widlowski, R. A. Fournier, A model for deriving voxel-level tree leaf area density estimates from ground-based LiDAR, Environmental Modelling and Software 51 (C) (2014) 184–189.

[38] J.-F. Côté, R. A. Fournier, R. Egli, An architectural model of trees to estimate forest structural attributes using terrestrial LiDAR, Environmental Modelling and Software 26 (6) (2011) 761–777.

[39] S. Martinuzzi, L. A. Vierling, W. A. Gould, M. J. Falkowski, J. S. Evans, A. T. Hudak, K. T. Vierling, Mapping snags and understory shrubs for a LiDAR-based assessment of wildlife habitat suitability, Remote Sensing of Environment 113 (12) (2009) 2533–2546.

[40] M. Abily, N. Bertrand, O. Delestre, P. Gourbesville, C.-M. Duluc, Spatial Global Sensitivity Analysis of High Resolution classified topographic data use in 2D urban flood modelling, Environmental Modelling and Software 77 (C) (2016) 183–195.

[41] P. Costabile, F. Macchione, Enhancing river model set-up for 2-D dynamic flood modelling, Environmental Modelling and Software 67 (C) (2015) 89–107.

[42] B. E. Romero, T. L. Bray, Analytical applications of fine-scale terrestrial lidar at the imperial Inca site of Caranqui, northern highland Ecuador, World Archaeology 46 (1) (2014) 25–42.

[43] R. Wang, 3D building modeling using images and LiDAR: a review, International Journal of Image and Data Fusion 4 (4) (2013) 273–292.

[44] O. Kreylos, G. Bawden, T. Bernardin, M. I. Billen, E. S. Cowgill, R. D. Gold, B. Hamann, M. Jadamec, L. H. Kellogg, O. G. Staadt, D. Y. Sumner, Enabling scientific workflows in virtual reality, Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications (2006) 155–162.

[45] R. B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL)., IEEE International Conference on Robotics and Automation ICRA (2011) 1–4.

[46] W. Schroeder, K. M. Martin, W. E. Lorensen, The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, The visualization toolkit (2nd ed.): an object-oriented approach to 3D graphics.

[47] J. Blanchette, M. Summerfield, C++ GUI Programming with Qt 4, 2nd Edition., Pearson Education.

[48] B. Karlsson, Beyond the C++ standard library: an introduction to boost, Pearson Education (2005).

[49] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, M. Beetz, Towards 3D Point cloud based object maps for household environments, Robotics and Autonomous Systems 56 (11) (2008) 927–941.

[50] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, C. T. Silva, Computing and Rendering Point Set Surfaces., IEEE Transactions on visualization and computer graphics 9 (1) (2003) 3–15.

[51] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517.

[52] T. Rabbani, F. van den Heuvel, G. Vosselman, Segmentation of point clouds using smoothness constraint, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (5) (2006) 248–253.

[53] M. Duckham, L. Kulik, M. Worboys, A. Galton, Efficient generation of simple polygons for characterizing the shape of a set of points in the plane, Pattern Recognition 41 (10) (2008) 3224–3236.

[54] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proceedings of the fourth Eurographics symposium on Geometry processing, Eurographics Association, 2006.

[55] Z. C. Marton, R. B. Rusu, M. Beetz, On fast surface reconstruction methods for large and noisy point clouds., IEEE International Conference on Robotics and Automation (2009) 3218–3223.

[56] H. Hoppe, New quadric metric for simplifiying meshes with appearance attributes, in: Proceedings of the conference on Visualization '99: celebrating ten years, IEEE Computer Society Press, 1999, pp. 59–66.

[57] P. Lindstrom, Out-of-core simplification of large polygonal models, in: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 259–262.

[58] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, W. Stuetzle, Piecewise smooth surface reconstruction, in: Proceedings of the 21st annual conference on Computer graphics and interactive techniques, ACM, 1994, pp. 295–302.

[59] D. Zorin, P. Schröder, W. Sweldens, Interpolating Subdivision for meshes with arbitrary topology, in: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM, 1996, pp. 189–192.

[60] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, H. P. Seidel, Laplacian surface editing, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM, 2004, pp. 175–184.

Appendix A. Supplementary Material

Habitat3D offers a variety of different filtering, clustering, segmentation and meshing routines. All these routines can be arbitrarily assembled to pre-defined pipelines operating on either subsets (i.e. clusters) or complete clouds. A list of all available algorithms is given here. Furthermore, some implementation details of Habitat3D are given.

Appendix A.1. Overview of the Implementation

As discussed in the introduction, a key practical requirement to model a large scale 3D natural environment from dozens of scans is to be able to inspect and process the data in a trial and error fashion. We identified four key requirements for the software:

1. An interface offering all necessary processing steps (including filtering, clustering and meshing), as described in the following sections.
2. 3D visual feedback including cloud visualization and (intermediate) results.
3. Batch- and recipe-based processing once an appropriate pipeline has been found.
4. Reliability and extensibility of the framework is mandatory to make it useful for a wide community.

Our platform-independent and open-source framework *Habitat3D* is implemented in C++ using well-established libraries. Most of the point cloud processing steps are implemented by utilizing the Point Cloud Library (PCL v1.7.2) [45] and additional mesh processing procedures are build based on the Visualization Toolkit (VTK v3.6.0) [46]. The interface is implemented using Qt (v4.8.7) [47] and BOOST (v1.58.0) [48] is used for performance reasons. To ensure reliability and enable extensibility our framework as far as possible integrates (rather than reimplements) established code libraries. Furthermore, we extended the existing libraries by several additional routines, more general data types (i.e. abstraction layers) and implemented a generalised template-based interface to call all filtering operations using virtual function calls. This enables an easy plug-in integration for future extensions and reduces the programming effort.

In order to interact with the data, our framework features the following general processing steps:

- Load, filter and inspect multiple clouds simultaneously

- Visualise the topology, normals, clusters and polygon meshes
- Merge and split clouds (or cloud clusters)
- Save clouds, clusters, normals and meshes in standard formats

Most importantly, all processing steps (besides merging / splitting clouds or clusters) can be applied to an arbitrary number of clouds (i.e. batch processing) either directly or in a recipe-like fashion. Assuming an appropriate order and parameter settings of processing steps has been determined, all clouds can be loaded and the sequence of steps can be programmed upfront. After starting the batch job the user is informed about the progress and intermediate results of all clouds can be saved on-the-fly. Furthermore, the user is informed about the (changes in) cloud / polygon size and a rudimentary undo function can be used for trial and error testing. All resultant objects (i.e. point clouds, normal clouds, segmentation results and meshes) can be saved. Several well-known formats are supported, namely *ply*, *obj*, *vtk* and *pcd*.

In order to provide a first quality measure of the calculated meshes we implemented the triangle quality measure which is based on the following formula: $q = \frac{4a\sqrt{3}}{h_1^2+h_2^2+h_3^2}$. a is the area and h_1, h_2 and h_3 specify the side lengths of the triangle. We calculate the mean quality measure q of all triangles of the resultant mesh and output the value on the terminal.

Appendix A.2. Software Availability, Requirements & Affiliation

Both the meshing framework (*Habitat3D*) and the resultant 3D world models of the animal habitats are available as open source downloads at the Ant Navigation Challenge website. There are no hardware or software requirements and the source code and compiled binary both require approximately 3Mb each. The accompanying cmake file was tested for Linux and MacOS X (10.9 or higher). The resultant Canberra reconstruction requires 470Mb and the Seville dataset requires 941Mb. Readme files highlighting all used processing steps and parameters are also included in the respective folders. It was implemented by Benjamin Risse at the University of Edinburgh (Institute for Action Perception and Behaviour; contact: brisse@inf.ed.ac.uk).

Appendix A.3. Processing Routines

All routines are described with special attention to their usability for natural outdoor scene modelling.

Appendix A.3.1. Point Cloud Filtering

Filtering options include noise / outlier removal, up- and down-sampling and conditional filtering (i.e. position or colour).

Radius-based Outlier Removal. Noise during scanning or inaccuracies in registration can cause small sets of points which do not correspond to any objects in the world, e.g. individual points in the sky, or from moving objects. These measurements can be characterised as outliers by setting a lower threshold for the number of neighbouring points within a certain radius.

Statistical Outlier Removal. It can be difficult to set the above mentioned threshold manually, since it should be set relative to the density of the given cloud $\mathcal{C}$ or clusters $c \in \mathcal{I}$. Instead, an appropriate threshold can be generated automatically by calculating the mean distance and standard deviation of all distances within a fixed number of neighbouring points [49]. Removing these outliers sharpens the object surfaces of natural structures such as plants.

Down-Sampling. Merging registered clouds with huge overlapping regions can cause massive redundancy, and inaccuracies in registration can cause slight displacement of points belonging to the same (*rigid*) object. Thus, a down-sampling strategy which tries to preserves the underlying surface is important. 3D voxel-grid based down-sampling can be used in which each grid is represented by a 3D box with user specified dimensions. All points within a grid are replaced by the centroid of the box resulting in evenly distributed points across the underlying surface.

Re-Sampling & Up-Sampling. Scanner blind-spots (immediately below the rig) and occlusions caused by foreground objects result in gaps in the data. To close these gaps, to smooth the clouds or to improve the normal estimation moving least squares (MLS) up-sampling can be used. By assuming that a given set of points is defining a surface, an MLS surface is defined as the points projecting on themselves [50]. The surface can be locally approximated by either a local plane, a random uniform density or a voxel grid and is specified by a search radius and the polynomial order.

Conditional Filtering. Points with certain spatial properties along the x-, y- or z-axis can be excluded using spatial conditional filtering. If for example the maximum height

of all objects in the scene is known, every point above this value can be excluded by introducing an upper limit along the z-axis. Similarly, upper and lower limits can be defined for red, green and blue colour values to exclude or extract objects with certain colour appearances.

Appendix A.3.2. Normal Extraction

A surface normal can be calculated for each point $p \in \mathcal{C}$ and is a vector perpendicular to the approximated tangent plane to the underlying surface at this point p. If normals are used as a feature to disambiguate ground and vegetation, non-smoothed surface perpendiculars are recommended. To approximate the underlying surface either a fixed number of nearest neighbour points or a maximal search distance can be specified. A k-d tree representation is used in order to speed up neighbouring point identification [51]. Normals can be estimated relative to the origin of the world coordinate system (the global $(0, 0, 0)$-point) or relative to the centroid of a cloud-subset: Normals belonging to the ground should be calculated given the global origin and normals for the respective plants should be relative to the plant cluster centroid (normal origin can affect subsequent meshing).

Meshing strategies like triangle meshing or Poisson surface reconstruction utilize vertex normals to generate the surface. As a consequence the smoothness of the surface relies on the smoothness of the normals. We added a smooth normal extraction possibility by applying moving least squares up-/re-sampling before normal calculation (see Appendix A.3.1).

Appendix A.3.3. Point Cloud Clustering & Segmentation

To identify points belonging to the same object we implemented several clustering and segmentation strategies. Whilst some of the algorithms are more appropriate to segment individual objects, others are helpful to identify global structures, e.g., finding all ground points $\mathcal{G} \subset \mathcal{C}$, and assuming the disjoint set $\mathcal{V} = \mathcal{C} \setminus \mathcal{G} = \{p | p \in \mathcal{C} \wedge p \notin \mathcal{G}\}$ includes all vegetation points. Besides extracting semantically meaningful subsets, splitting the cloud $\mathcal{C}$ into clusters $c \in \mathcal{I}$ is mandatory to speed up processing and can also be necessary to define regions based on their distance to the sensor (the point density decreases with the distance to the scanner).

Distance-based Clustering. Assuming that points belonging to the same object form a compact distribution, these points can be clustered into semantic entities by using L2 Euclidean distance cluster extraction (in (x, y, z) direction): By using a Kd-tree, nearest neighbours for each point are determined. Subsequently clusters are constructed by adding all nearest neighbours within a sphere radius (here called cluster tolerance) that have not been processed yet.

Concentric Clustering. Since terrestrial laser scanners sample nearby surfaces more densely than distal surfaces, the framework also provides a circular clustering procedure using a user specified centre point (x, y) and radius for the maximally allowed Euclidean distance to this point.

Iterative Chunk Split Clustering. The global cloud for a habitat can comprise billions of points which is far to big to be processed by most of the algorithms, but down- or sub-sampling could collapse or delete thin (but significant) 3D volumes. In these situations iterative chunk split clustering can be used to separate the cloud $\mathcal{C}$ into subsets $c \subset \mathcal{C}$ with $|c| \ll |\mathcal{C}|$ $(\bigcup_i c_i = \mathcal{C})$. The maximal size of each cluster can be controlled by a reduction enforcement value specifying the maximal allowed size of a cluster. Each cluster is recursively quartered in (x, y)-directions until the size is below or equal to this maximal size or if it covers more than a certain percentage of the total area. Note that this splits can separate connected components (e.g. a branch from a tree) so that merging after processing the chunks is recommended. In case splitting needs to be avoided distance-based clustering with a small neighbourhood radius can be used to generate compact super-voxel.

Progressive Morphological Filtering. Progressive morphological filtering was initially introduced to remove non-ground measurements from airborne LiDAR measurements [28]. The filter gradually increases the window size of an opening filter and uses elevation difference thresholds to identify ground points. Depending on the size of the modelled area the window size can either be increased linearly or exponentially to reduce the number of iterations used for filtering. Since both the allowed difference in elevation and change of window size of the opening operation can be controlled by respective parameters, this filter can be tuned to be either very specific (i.e. high true negative rate [true positive ground points divided by all ground points] or correct rejection rate) or very sensitive (i.e. high true positive rate [true positive vegetation points divided

by all vegetation points], recall or hit rate). We use this method to identify vegetation points in terrestrial scans and cluster them into $\mathcal{V}$ and $\mathcal{G}$. Thus, instead of removing non-ground points we iteratively add these *removed* points to $\mathcal{V}$. By setting the parameters to get high specificity we ensure a very close-to-the-ground vegetation segmentation. Progressive morphological filtering rejects vegetation points (i.e. negative detections) and high specificity maximises a true negative classification.

Region Growing Segmentation. A second ground segmentation strategy in *Habitat3D* is region growing segmentation. Region growing segmentation utilises surface normals in order to identify ground points. Assuming a more or less flat ground containing only smooth slopes, the general curvature and distributions of normals are distinctive for regions of the point cloud belonging to the ground plane. Consequently, region growing segmentation utilises a smoothness constraint to find smoothly connected areas within the cloud [52]. The connectivity is specified by the number of neighbours and the smoothness constraint is achieved by thresholding normals above a certain value and restricting the curvature along the underlying ground surface. A combination of region growing segmentation after progressive morphological filtering yielded best results to identify ground points in our datasets.

Concave Cloud Hull Extraction. Laser scanners strobe the outer boundary of the objects within the scanning radius. However, due to inaccuracies of the scanner and merging several scans into a single global cloud, points are also *inside* the outer boundary. To extract the outer boundary of the underlying objects, concave hulls (also called alpha shapes) can be extracted [53]. In contrast to convex hull, which is uniquely defined by the set of points minimizing the area covering all given points without having any angle that exceed 180° between edges, a concave hull also minimizes the area of the resultant shape allowing any angle (resulting in convex and concave structures along the boundary).

Appendix A.3.4. Meshing

The actual transformation from point clouds into 3-dimensional polyhedral objects is called meshing. Meshes are a collection of vertices which are connected by edges to build closed faces. Combined faces define polygons which are used to describe surfaces. Meshes are a generalised digital data structures of triangulated irregular networks (TIN).

Poisson Surface Reconstruction. All above listed ground meshing requirements are addressed using the Poisson Surface Reconstruction method in which the reconstruction is formulated as a spatial Poisson problem. The resolution of the reconstruction can be controlled by the depth of the underlying adaptive octree. High tree depths result in higher-resolution functions used to fit the indicator function (i.e. a vector field that is zero almost everywhere except at points near the surface) so that the model captures finer details, whereas low depths result in a smoother surface. Since all ground points $\mathcal{G}$ are meshed at once the reconstruction is more resilient to noise and gaps. Poisson surface reconstruction utilises the relationship between oriented points (i.e. points and normals) sampled from a surface and the indicator function of the model [54]. Consequently, the smoothness of the resultant mesh relies crucially on the normal directions extracted prior to meshing.

Concave Hull Meshing. Concave hulls are directly estimated from the plant clusters [53], making this procedure more general for different types of plants and independent to surface normals but at the cost of smoothness. The general principle is equivalent to the concave cloud hull extraction explained in Appendix A.3.3. To avoid over-fitting a parameter α is set to limit the size of the resultant hull segments (the smaller the more detailed the alpha-shape), so that α controls which neighbouring points are connected. A facet is accepted if the distance from any point to the centre of the voronoi cell (i.e. the facet centre) is smaller than α. Setting α relative to the minimal distance between natural structures (e.g. between two blossoms of a single flower) the resultant plant models have closed surfaces without merging nearby but different structures. Note that the merged plant clusters can directly be meshed via concave hull meshing: neither particular pre-processing nor normal extraction is necessary. However, since the hull spans a triangle network covering the outermost points, the resultant mesh will have sharp edges.

Greedy Triangulation Meshing. As an alternative to the above listed meshing strategies, greedy triangulation meshing can be used to generate general triangle meshes. Since these meshes rely on local neighbourhoods and surface normals, local smoothing is recommended before meshing. In principle the mesh is generated by successively connecting points (i.e. add triangles) to grow the surface incrementally. The neighbourhood for a point p is determined by a user specified search radius and a tangent

plane is projected through these points along p's normal [55]. The smoothness of the resultant surface can be controlled by a minimum and maximum angles in each triangle and a maximum surface angle (i.e. two points can only be connected if the difference of their normals is below the maximum).

Appendix A.3.5. Mesh Filtering

The mesh appearance and complexity can be optimised by applying mesh filtering techniques. In our framework, the natural environments are extracted from raw point clouds so that processing of the clouds before meshing usually yields better results than filtering the resultant polygon models. However, to offer a complete processing framework we added options for four mesh filtering strategies to reduce, refine or smooth the mesh.

Quadric Mesh Decimation. Quadric mesh decimation reduces the number of faces based on repeated edge collapsing by utilising priority queue-based costs [56]. It continues until a desired reduction level is reached or topological constraints prevent further reduction.

Quadric Clustering. Since quadric mesh decimation is restricted by topological constraints, it sometimes does not yield satisfactory reduction rates. We therefore added quadric clustering as an alternative which can be used to enforce the face reduction of the mesh. Note that it can cause disconnected meshes (i.e. mesh topology is not preserved) but allows simplification of arbitrary complex datasets in linear time. Quadric clustering bins the 3D space by accumulating the quadric surface of each face and extracts the optimal vertex position for each bin based on the accumulated surfaces [57]. The number of divisions can either be calculated automatically or manually set for all 3 dimensions.

Mesh Subdivision. Besides decimating a mesh to reduce the number of faces, refinement or subdivision can be used to increase the number of faces. The *Habitat3D* framework offers three well-known subdivision schemes: linear, loop and butterfly subdivision. The linear scheme divides each face into four new faces so that the resultant polygon mesh has a smoother surfaces but at the costs of an increase in memory required. In the loop scheme, each face is also subdivided into four sub-faces but in addition the vertices of the refined mesh are then positioned using a weighted average

of the vertices in the unrefined mesh, thus approximating the initial geometry [58]. Instead of approximating (i.e. generating new control points), butterfly refinement can be used to interpolate old and new control points on the original surface mesh (Butterfly subdivision is named after the shape of the scheme) [59].

Laplacian Mesh Smoothing. Similar to point cloud smoothing, mesh smoothing results in a less sharp surface given more evenly distributed vertices. We integrated Laplacian mesh smoothing into our framework which uses local informations (i.e. 3D position of neighbouring vertices) to extract a new position. The coordinates of a vertex are modified according to an average of the neighbouring vertex positions with respect to a relaxation factor controlling the amount of displacement [60].

Appendix A.4. Parameter Discussion

The used meshing recipes (pipelines as well as all specified parameters) are available at insectvision.org. Here we discuss the main parameters and how different parameter settings influence the resultant models. The dimensions of the point clouds are determined by the used scanning hardware. The overall shape of the ground model is mainly controlled by the parameters specified for (a) the progressive morphological filtering; (b) the voxelgrid based down-sampling; (c) the region growing segmentation; and (d) the Poisson surface reconstruction. During progressive morphological filtering the slope and maximal distance between points can be specified to identify jumps in elevation. Since we fine-tune the shape of the ground during latter meshing comparatively large slope and distance values can be used. Down-sampling specifies the *resolution* of the final model since the resultant subset is the source for the resultant mesh. As explained in the main text, the most crucial parameters are set during region growing segmentation and meshing since these two stages crucially depend on each other (c.f. Section 2.4.2). If complex ground topologies need to be preserved the smoothness and curvature threshold have to be comparatively high (e.g. 18° and 6° respectively). Finally, the octree depth of the Poisson surface reconstruction needs to reflect the desired complexity of the surface (e.g. 6 for approximations and 10 for high resolution reconstructions).

In order to model vegetation we recommend to try a reconstruction without any sub-sampling to avoid fragmented plants. Instead statistical outlier removal with a comparatively small search neighbourhood (e.g. 50 points) can be used to sharpen

the clouds. Individual plant cluster should be extracted based on the density and size of the plants in the respective scene: We recommend to start with larger cluster tolerances (> 0.5) and inspect the results using Habitat3D. If the vegetation appears merged, smaller values can be chosen; we achieved good results with a tolerance of 0.3. Note that the minimal cluster size should be bigger than or equal to the search neighbourhood of the statistical outlier removal since no smaller clusters will exist after this procedure. Finally the α value of the concave hulls specifies the details of the plants. Given an $\alpha = 0.1$ very high polygon meshes will be produced. In contrast $\alpha > 0.5$ might result in oversimplifications of the underlying topology (e.g. merging neighbouring leafs). Again, we recommend to start with larger α values which should be below the cluster tolerance to capture fine details of the vegetation. Given plants in very close proximity to each other, like in dense forests (e.g. closed canopy), very small cluster tolerances are recommended to approximate individual plant identities. However, a correct segmentation cannot be achieved given highly overlapping plant structures.